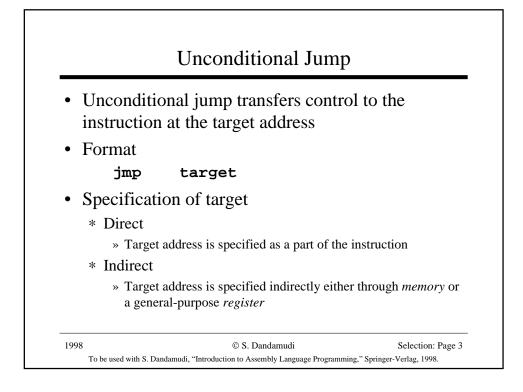
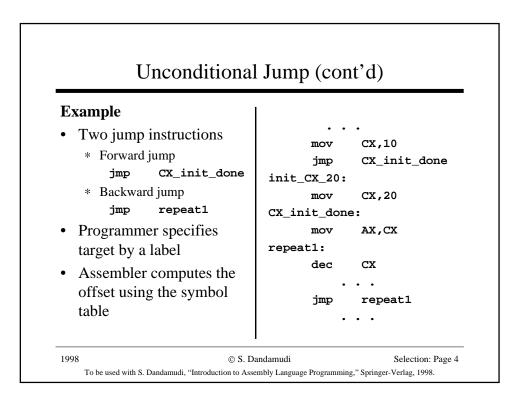
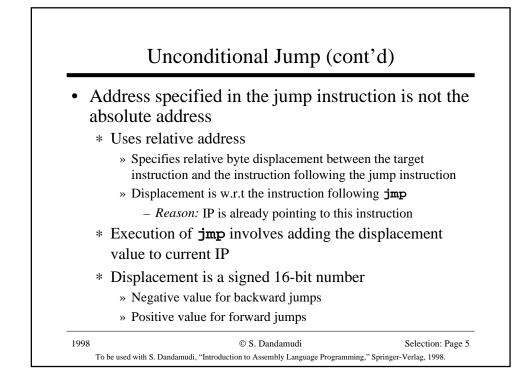


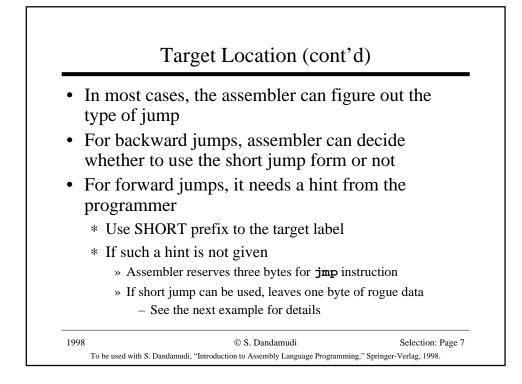
Unconditional jump Compare instruction Conditional jumps * Single flags * Unsigned comparisons * Signed comparisons Loop instructions Implementing high-level language decision structures * Selection structures * Iteration structures	<ul> <li>Illustrative examples</li> <li>Indirect jumps <ul> <li>Multiway conditional statements</li> </ul> </li> <li>Logical expression evaluation <ul> <li>Full evaluation</li> <li>Partial evaluation</li> </ul> </li> <li>Performance: Logical expression evaluation <ul> <li>Partial vs. full evaluation</li> </ul> </li> </ul>



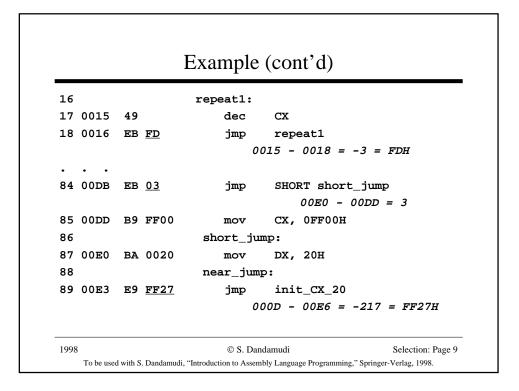


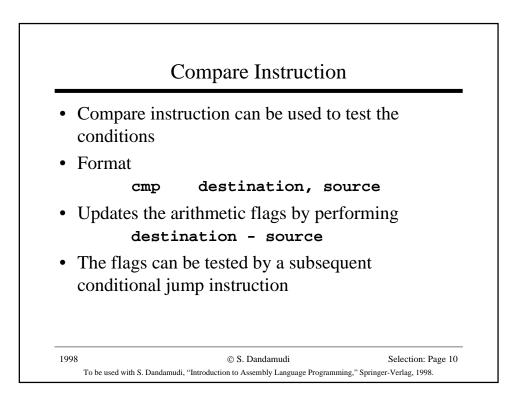


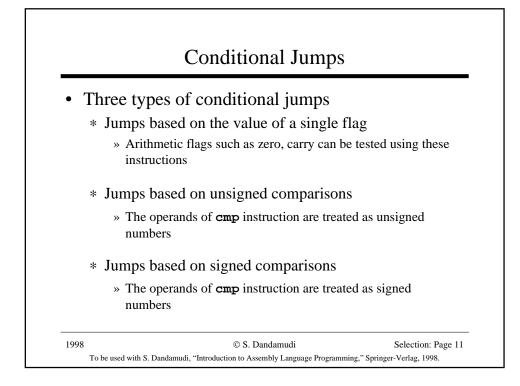
	Target Location
	<pre>nter-segment jump Target is in another segment CS := target-segment (2 bytes) IP := target-offset (2 bytes) Celled far immer (needs fine bytes to encode imm)</pre>
	<ul> <li>» Called <i>far jumps</i> (needs five bytes to encode jmp)</li> <li>ntra-segment jumps</li> <li>Target is in the same segment</li> <li>IP := IP + relative-displacement (1 or 2 bytes)</li> </ul>
	<ul> <li>Uses 1-byte displacement if target is within -128 to +127</li> <li>» Called <i>short jumps</i> (needs two bytes to encode jmp)</li> <li>If target is outside this range, uses 2-byte displacement</li> <li>» Called <i>near jumps</i> (needs three bytes to encode jmp)</li> </ul>
1998 T	© S. Dandamudi Selection: Page 6 o be used with S. Dandamudi, "Introduction to Assembly Language Programming," Springer-Verlag, 1998.



<pre>8 0005 EB <u>0C</u> jmp SHORT CX_init_done</pre>
0013 - 0007 = 0C 9 0007 B9 000A mov CX,10 10 000A EB 07 90 jmp CX_init_done rogue byte 0013 - 000D = 07 11 init_CX_20: 12 000D B9 0014 mov CX,20 13 0010 E9 00D0 jmp near_jump
10 000A EB <u>07</u> 90 jmp CX_init_done rogue byte 0013 - 000D = 07 11 init_CX_20: 12 000D B9 0014 mov CX,20 13 0010 E9 <u>00D0</u> jmp near_jump
rogue byte 0013 - 000D = 07 11 init_CX_20: 12 000D B9 0014 mov CX,20 13 0010 E9 00D0 jmp near_jump
11       init_CX_20:         12 000D B9 0014       mov       CX,20         13 0010 E9 00D0       jmp       near_jump
12 000D B9 0014 mov CX,20 13 0010 E9 <u>00D0</u> jmp near_jump
13 0010 E9 <u>00D0</u> jmp near_jump
0.0E3 - 0.013 = D0
14 CX_init_done:
15 0013 8B C1 mov AX,CX







<b>Festing</b> fo	or zero	
jz	jump if zero	jumps if ZF = 1
je	jump if equal	jumps if $ZF = 1$
jnz	jump if not zero	jumps if $ZF = 0$
jne	jump if not equal	jumps if $ZF = 0$
jcxz	jump if $CX = 0$	jumps if $CX = 0$
		(Flags are not tested)

Jumps Based on Single Flags (cont'd)			
Testing f	or carry		
jc	jump if carry	jumps if $CF = 1$	
jnc	jump if no carry	jumps if $CF = 0$	
Testing fo	or overflow		
jo	jump if overflow	jumps if $OF = 1$	
jno	jump if no overflow	jumps if OF = 0	
Testing f	or sign		
js	jump if negative	jumps if $SF = 1$	
jns	jump if not negative	jumps if $SF = 0$	
1998	© S. Dandamudi	Selection: Page 13	

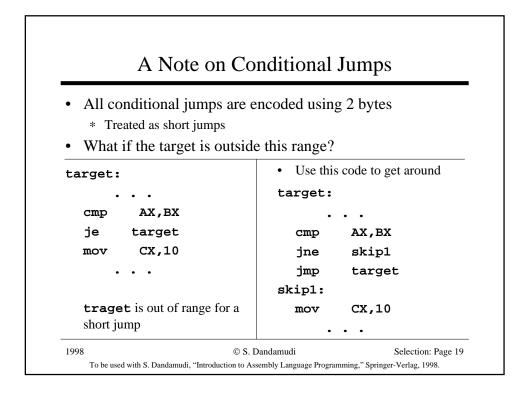
<b>Festing</b> f	or parity	
qt	jump if parity	jumps if PF = 1
jpe	jump if parity is even	jumps if PF = 1
jnp	jump if not parity	jumps if $PF = 0$
јро	jump if parity is odd	jumps if PF = 0

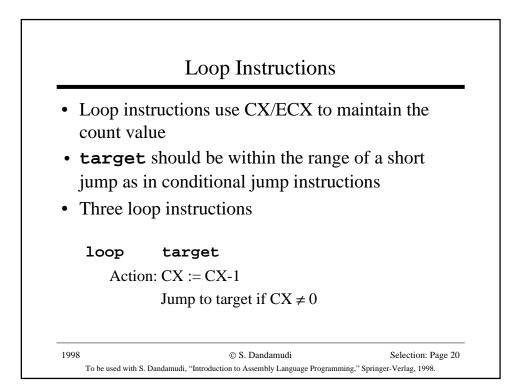
Mnemonic	Meaning	Condition
je	jump if equal	ZF = 1
jz	jump if zero	ZF = 1
jne	jump if not equal	ZF = 0
jnz	jump if not zero	ZF = 0
ja	jump if above	CF = ZF = 0
jnbe	jump if not below or equal	CF = ZF = 0

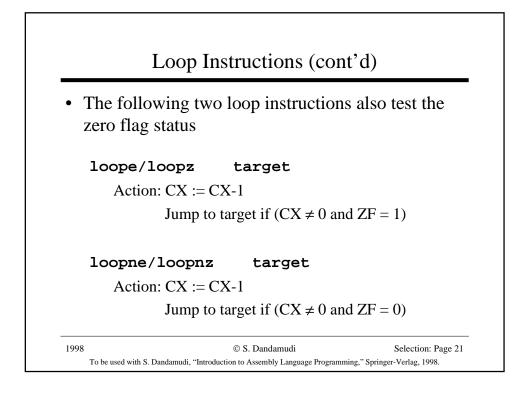
Jumps Based on Unsigned Comparisons			
Mnemonic	Meaning	Condition	
jae	jump if above or equal	CF = 0	
jnb	jump if not below	CF = 0	
jb	jump if below	CF = 1	
jnae	jump if not above or equal	CF = 1	
jbe	jump if below or equal	CF=1 or ZF=1	
jna	jump if not above	CF=1 or ZF=1	

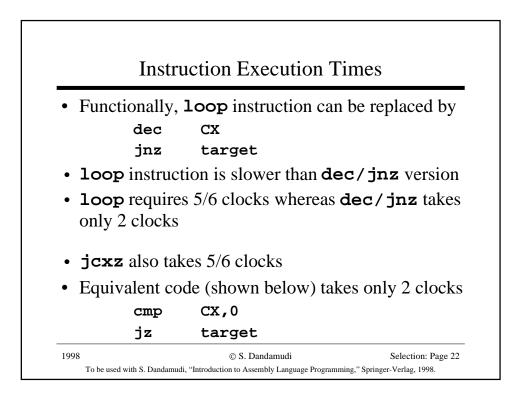
Meaning	
	Condition
jump if equal	ZF = 1
jump if zero	ZF = 1
jump if not equal	ZF = 0
jump if not zero	ZF = 0
jump if greater	ZF=0 & SF=OF
jump if not less or equal	ZF=0 & SF=OF
	jump if zero jump if not equal jump if not zero jump if greater jump if not less

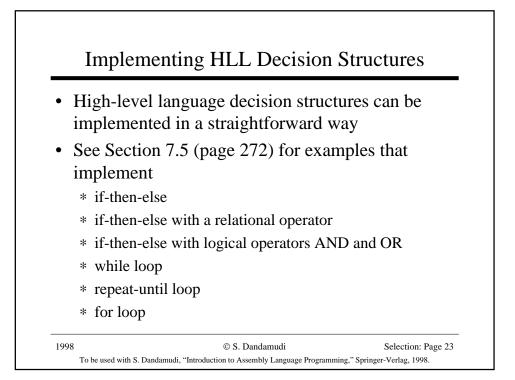
Mnemonic	Meaning	Condition
jge	jump if greater or equal	SF = OF
jnl	jump if not less	SF = OF
jl	jump if less	SF ≠ OF
jnge	jump if not greate or equal	er SF≠OF
jle	jump if less or equal	$ZF=1 \text{ or } SF \neq C$
jng	jump if not greate	er ZF=1 or SF $\neq$ C

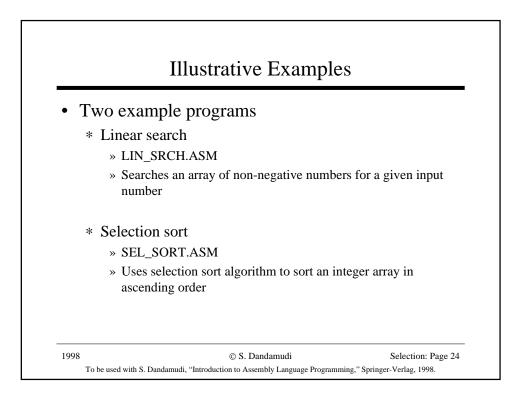


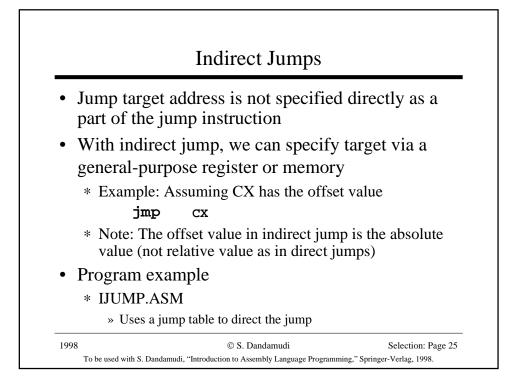


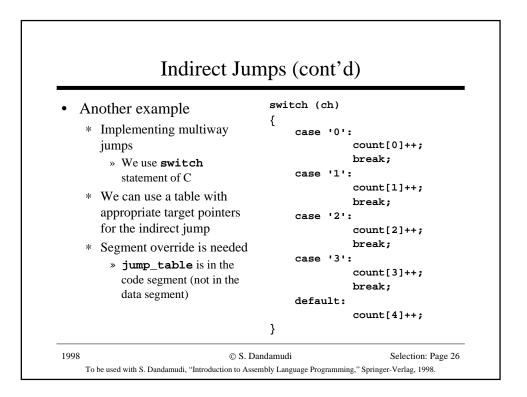












	Indirect Jun	nps (cont'd)
_main mov cbw sub mov cmp ja shl jmp	PROC NEAR AL, ch AX, 48 ;48 =`0' BX, AX BX, 3 default BX,1 ;BX:= BX*2 WORD PTR CS:jump_table[BX]	<pre>case_2: inc WORD PTR [BP-6] jmp SHORT end_switch case_3: inc WORD PTR [BP-4] jmp SHORT end_switch default: inc WORD PTR [BP-2] end_switch:  _main ENDP</pre>
case_0: inc jmp case_1: inc jmp	WORD PTR [BP-10] SHORT end_switch WORD PTR [BP-8] SHORT end_switch	jump_table LABEL WORD DW case_0 DW case_1 DW case_2 DW case_3 
1998 To be used		andamudi Selection: Page 27 embly Language Programming," Springer-Verlag, 1998.

